Attorney's Docket No.: 42P17030

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application for:	
Zhong-Ning Cai, et al.	Examiner: Patel, Kaushikkumar M.
Application No.: 10/611,619	Art Group: 2188
Filed: June 30, 2003	
For: **APPARATUS AND METHOD FOR AN ADAPTIVE MULTIPLE LINE PREFETCHER**	

DECLARATION PURSUANT TO 37 C.F.R. §1.131

Mail Stop Amendment
Commissioner for Patents
P. O. 1450
Alexandria, VA 22313-1450

Dear Sir:

I, Zhong-Ning Cai, hereby declare that:

1. I am a citizen of the United States of America.
2. I currently reside at 4120 Canal Road, Lake Oswego, Oregon 97034.
3. I am currently an employee of Intel Corporation in Santa Clara, California.
4. I have been an employee of Intel Corporation since 1997.
5. My current title at Intel Corporation is Principal Engineer.
6. I am a co-inventor of the above-identified patent application.

7. I have reviewed the 2004/0148470 patent application publication of Jurgen Schulz ("Schulz"), which was filed on January 29, 2003. The Examiner cited Schulz against the claims of the above-identified application.

8. The invention disclosed and claimed in the above-identified patent application was conceived in the United States of America at least as early as January 19, 2003, as evidenced by the document "CFB Second Sector Prefetch" (a copy of which is attached herein). This document was reduced to writing internally within Intel Corporation at least as early as the date on the document; i.e., January 19, 2003. This document demonstrates conception of the claimed invention of the instant application. Intel Corporation Invention Disclosing having ID #30489 (a copy of which is attached herein) was prepared based on the "CFB Second Sector Prefetch" document at least as early as the date on the date on the document; i.e., February 10, 2003. Between at least January 2003 and its constructive reduction to practice by the filing of the above-captioned patent application on June 30, 2003, I directed simulations and various testing in a diligent effort to reduce the invention to practice. Therefore, the conception and diligence towards reduction to practice of the invention disclosed and claimed in the above-identified patent application occurred prior to the filing date of Schulz.

9. The documents provided herewith are confidential. It is Intel Corporation's practice to maintain in secrecy all confidential documents. I believe that the documents have at all times prior to the filing date of the above-captioned application been maintained in a confidential manner.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States

Code, and that such willful false statements may jeopardize the validity of the above-identified application or any patent issued thereon.

Respectfully submitted,

Dated: July 26, 2006

Zhong-Ning Cai

Full Name:

Zhong-Ning Cai
Citizenship: United States of America
Residence: 4120 Canal Road
Lake Oswego, Oregon 97034

CFB Second Sector Pre-fetch

Will Auld

01-19-03

Events and Actions:

On BRL, BRIL, ? transactions the PREFETCH_OK bit is checked, if set, the address from the transaction is converted into what would be the other sector of the core's uL1 line. This address is looked up in the CFB tags. If the line is present, nothing else needs to be done. If it is not present, check the victim buffer tags and select the victim way. If the victim way has a valid entry, back invalidate and move the data to the victim buffer. Start and complete a load for the second sector from either the victim buffer or the eFSB...

PREFETCH_OK = ENABLE_CFB_PREFETCH &
PREFETCH_LOAD_ACCEPTABLE

CFB_PREFETCH_LESS_EQ is the programmable acceptable IOQ Depth for continued second sector pre-fetch.

ENABLE_CFB_PREFETCH is the programmable enable/disable switch.

PREFETCH_LOAD_ACCEPTABLE is updated ever X, say 1024, eFSB cycles based on the average eIOQ depth. The average eIOQ depth is estimated by the following procedure: Each cycle the depth of the eIOQ is added to a 14-bit counter.[1] Every 1024 cycles this 14-bit counter's top 4 bits will represent the integer floor approximation of the average IOQ Depth over the last 1024 eFSB cycles. So PREFETCH_LOAD_ACCEPTABLE = top 4 bits <= CFB_PREFETCH_LESS_EQ. After PREFETCH_LOAD_ACCEPTABLE is updated the 14 bit counter is cleared and can resume its counting. (It is OK if this takes a few clocks the extra cycles can be counted as part of the 1024 cycles without loss of accuracy)

Justification for throttling mechanism:

Originally I thought we should base the pre-fetch throttling on both the eFSB latency and the core's CPI. A running average for the eFSB latency can be derived by dividing the 14-bit counter with the number of eFSB transactions that occurred in the last 1024 eFSB cycles. But the core's CPI is not readily available. After some consideration, it looks like using the eFSB IOQ depth will act as a good proxy for using both eFSB latency and utilization combined.

To investigate this further I used Little's law to estimate IOQ depth over a broad range of latency and utilization levels. Checking Little's law against a small set of runs I find that the Little's law over estimates IOQ depth slightly but is within 0.5 of the measured IOQ

[1] 1024 cycles * 12 IOQ depth = 12,288 is less than 16,384 which fits in a 14 bit counter.

depth. The following table gives the IOQ depth predicted by Little's law for utilizations

IOQ Depth Table based on Little's Law

		Util									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
latency	10	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
	15	0.75	1.50	2.25	3.00	3.75	4.50	5.25	6.00	6.75	7.50
	20	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
	25	1.25	2.50	3.75	5.00	6.25	7.50	8.75	10.00	11.25	12.50
	30	1.50	3.00	4.50	6.00	7.50	9.00	10.50	12.00	13.50	15.00

from 10% to 100% and latencies from 10 bus cycles to 30 bus cycles.

I believe that based on this table and some limited data (need to look at more data) setting CFB_PREFETCH_LESS_EQ to 5 would be the optimal value for Shasta/Gallatin. Now using the IOQ Depth projections setting CFB_PREFETCH_LESS_EQ to 5 allows 100% bus utilization with a latency of 10 bus clocks while at 20 bus clocks of latency pre-fetch would be turned off at 50% bus utilization. Given the inaccuracy of the prediction in the table above, the actual latency at which a system would turn off pre-fetch is a little higher for the utilization of 50%.

Note: if the system is operating close to the switch point, turning pre-fetch off could be enough to turn pre-fetch on and this might start an oscillation between this two states.

Simulation Metrics:

Metrics to be added to Willy for simulations studies.

CFB Second Sector Pre-fetch

1. Count of the number of pre-fetched lines
2. Count of the number of pre-fetched lines that cause a clean eviction
3. Count of the number of pre-fetched lines that cause a dirty eviction
4. Count of the number of pre-fetched lines that are accessed (actually used)
5. For demanded pre-fetched lines the average time between pre-fetch start and the demand access (Should include this for the core's pre-fetching as well)
6. Count of the number of eFSB cycles PREFETCH_LOAD_ACCEPTABLE clear (off)

Victim Buffer

1. Count the number of lines recalled from the victim buffer (Hit Count)
2. Average time a line stays in the victim buffer
3.

Cache (trace) study

1. Count of the total number of unique address requested (demanded address) for the run.
2. Count of the total number of unique address requested (demanded address) for a specific number of instructions executed (definable)
3. c (collision rate)
4. Fl2 and Fl2 probability of going from state 1 to 2 and 2 to 2.

Needed Changes:

1. Add low water mark as well as the high water mark.
2. Use more bits for better accuracy.
3.

INTEL INVENTION DISCLOSURE

ATTORNEY-CLIENT PRIVILEGED COMMUNICATION
located at http://legal.intel.com/patent/index.asp
Rev. 17 – January 2003

SOFTWARE/EPG/MRL

30489

DATE: 2-10-03

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application. **Invention Disclosure forms MUST be sent electronically via email to your manager/supervisor who should then forward with their approval to our email account "invention disclosure submission."** If you have any questions, please call **8-264-0444.**

Last Name: Cai	First Name: Zhong-Ning (George)		M.I.
Intel Phone Number: 503-264-2921	Intel Fax Number:		Mailstop: JF4-411
E-mail address: george.cai@intel.com			WWID: 10566878
Citizenship: USA	Are you a contractor?	Yes:	No: XXX
Home Address: 4120 Canal Rd			
City: Lake Oswego	State: OR	Zip: 97034	Country:
Corporate Level Group: EPG	Division: ODC		Subdivision: architecture
Supervisor: Gans Srinivasa	WWID: 10067570	M/S: JF4-411	Phone #: 503-712-4774

Last Name: Auld	First Name: William		M.I. G.
Intel Phone Number: (503) 712-8220	Intel Fax Number:		Mailstop: JF1-05
E-mail address: will.auld@intel.com			WWID: 10057407
Citizenship: USA	Are you a contractor?	Yes:	No: XXX
Home Address: 16823 NW Waterford Way			
City: Portland	State: OR	Zip: 97229	Country: USA
Corporate Level Group: EPG	Division: SSG		Subdivision: CSD
Supervisor: David C. Stewart	WWID: 10541683	M/S: JF1-05	Phone #: (503) 712-6013

Last Name: Gilbert	First Name: Jeffrey		M.I. D.
Intel Phone Number: 712-2070	Intel Fax Number: 712-2774		Mailstop: JF4-354
E-mail address: jeffrey.d.gilbert@intel.com			WWID: 10478966
Citizenship: USA	Are you a contractor?	Yes:	No: XXX
Home Address: 8740 SW Birchwood Rd			
City: Portland	State: OR	Zip: 97225	Country: USA
Corporate Level Group: DPG	Division: Architecture		Subdivision: Platform Arch
Supervisor: Gurbir Singh	WWID: 10039014	M/S: DP2-281	Phone #: 371-4180

(PROVIDE SAME INFORMATION AS ABOVE FOR EACH ADDITIONAL INVENTOR)

2. Title of Invention:

Dynamic Adaptive Multiple line Pre-fetcher (DAMP)

3. What technology/product/process (code name) does your invention relate to (be specific if you can)

Dual Core Potomac/Dual Core Cedar Mill

4. Include several key words to describe the technology area of the invention in addition to # 3 above:

Multiple Level Cache, System Bus Utilization, Memory Access Latency, Hit Ratio

5. Stage of development (i.e. % complete, simulations done, test chips if any, etc.):

33%, Initial general design done, working toward simulation

6a. Has a description of your invention been (or planned to be) published outside of Intel:

If YES, was the manuscript submitted for pre-publication approval through the Author Incentive Program:

If YES, please identify the publication and the date published:

Not at present

6b. Has your invention been used/sold or planned to be used/sold by Intel or others?

If YES, date it was sold or will be sold:

Going through the design process for Dual Core Potomac

6c. Is a SIG (special interest group) active in this technology?

If YES, name of SIG:
No

6d. If the invention is embodied in a semiconductor device, actual or anticipated date of tapeout?

Dual Core Potomac Expected tapeout date June 2004 and Dual Core Cedar Mill June 2005

6e. If the invention is software, actual or anticipated date of any beta tests outside Intel:

7. Was the invention conceived or constructed in collaboration with anyone other than an Intel blue badge employee or in performance of a project involving entities other than Intel (e.g. government, other companies, universities or consortia)? NO: XXX If YES, name of individual or entity:

8. Is this invention related to any other invention disclosure that you have recently submitted? If so, please give the title and inventors:
It is not related to any other.

****

PLEASE READ AND FOLLOW THE DIRECTIONS ON HOW TO WRITE A DESCRIPTION OF YOUR INVENTION

Try to limit your description to 2-3 pages
Do NOT attach a presentation, white paper, or specification
ANSWER ALL OF THE QUESTIONS BELOW

Please provide a description of the invention and include the following information:

1. **Describe in detail what the components of the invention are and how the invention works.**
2. **Describe advantage(s) of your invention over what is currently being done.**
3. **You MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.**
4. **Value of your invention to Intel (how will it be used?).**
5. **Explain how your invention is novel. If the technology itself is not new, explain what makes it different.**
6. **Identify the closest or most pertinent prior art that you are aware of.**
7. **Who is likely to want to use this invention or infringe the patent if one is obtained?**
8. **How would infringement be detected?**

HAVE YOUR MANAGER/SUPERVISOR READ AND FORWARD THIS DISCLOSURE ELECTRONICALLY VIA E-MAIL TO "INVENTION DISCLOSURE SUBMISSION"

BY APPROVING, YOUR MANAGER/SUPERVISOR IS ACKNOWLEDGING THAT THE DISCLOSURE HAS BEEN READ AND UNDERSTOOD, AND RECOMMENDS THAT THE DISCLOSURE AWARD BE PAID

Background:
A traditional instruction and/or data pre-fetching mechanism focuses on the requested address patterns. These pre-fetch mechanism designs / algorithms aim to accurately predict which memory lines will be requested in the future based on what has been requested recently. However, pre-fetching can rapidly increase the memory subsystem utilization. The relationship between system memory access latency and high memory subsystem utilization negatively impacts the pre-fetching mechanism's effectiveness. In some SMP systems (DP Prestonia, MP Gallatin), aggressive pre-fetching drives up the memory subsystem utilization, thereby increasing latency to the point that system performance is worse with pre-fetching than without it. This is an important performance issue for SMP and CMP systems.

Observation of the relationship between memory subsystem utilization and memory latency suggests that most memory subsystems experience a rather notable inflection point. Increasing memory utilization from near zero up to that inflection results in negligible increases in memory latency. However, after utilization reaches the inflection point, further utilization increases cause substantial and growing increases in memory latency. Stated another way, this observation partitions the cost of incremental memory utilization: the cost is low when the memory subsystem is operating below the inflection point and high when the memory subsystem is operating above the inflection point.

Invention:
The Dynamic Adaptive Multiple line Pre-fetcher (DAMP) is a pre-fetcher that can adaptively change pre-fetching depth (number of memory lines to be pre-fetched) according to the memory subsystem utilization. In this discussion below, it is important to note that the control variable is memory subsystem utilization rather than bus utilization. DAMP tracks the memory subsystem occupancy and uses the level of occupancy to decide the pre-fetch depth. By adjusting the pre-fetch depth, we can mitigate the negative performance impact while maintaining the positive aspects of pre-fetch operations.

The main idea of DAMP is to provide dynamic control logic that optimizes the performance of simple linear pre-fetching, memory throughput and memory latency. DAMP addresses key issues with current practice by adjusting memory subsystem demand to avoid negative performance impacts of being forced to always do pre-fetching or never do pre-fetching.

1) Hardware Pre-fetch enabled/disabled (STREAM gets a 33% boost but 4P TPC-C loses ~4%)
2) Second Sector enabled/disabled (1 and 2P TPC-C gets 4-6% boost but 4P TPC-C loses 2-10%)

The main cause of poor pre-fetch performance, when usage prediction is reasonably good, is the increase in memory latency for non-pre-fetch loads. On most systems the curve describing the relationship between memory subsystem throughput and latency has a fairly sharp knee. At low throughput levels a relatively large change in throughput will not have a significant impact on latency. However, at higher throughput levels, above the knee in the curve, a small change in throughput will cause a large increase in latency.

Despite the exponential-like relation between throughput and latency, using a system parameter not directly related to subsystem utilization – such as throughput or bus utilization – as the indication of when to enable, disable or adjust the level of pre-fetching has major drawbacks. For instance, it penalizes memory subsystems designed to sustain high throughput rates at reasonable performance with poorly designed subsystems that experience their inflection point at lower throughput rates. Rather, the use of memory subsystem occupancy allows us to target latency without having to actually track latency.

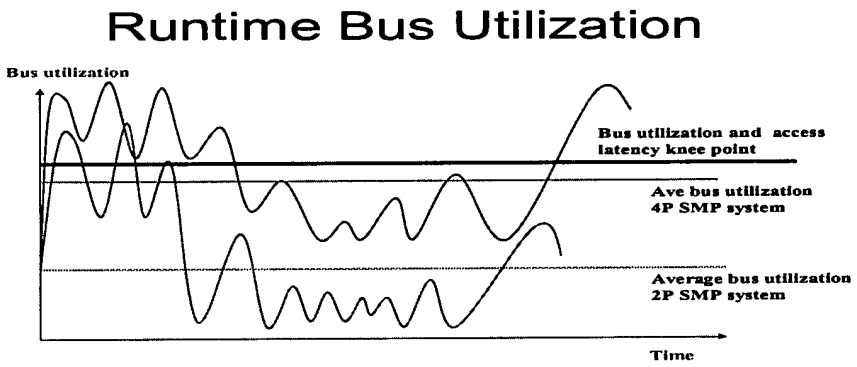
Justification for throttling based on occupancy:
Our goal in the use of a pre-fetching mechanism is to load data into the caches before the processor requests the data thereby lowering the effective latency for memory requests. Fundamental to the nature of pre-fetching is an increase in the memory subsystem throughput. This increase in throughput is not harmful in and of itself. Rather the increase in latency for those processor requests that still miss the cache is the problem. So basing the throttling mechanism on latency rather than throughput is important. While considering ways to monitor average latency, I realized that memory subsystem occupancy is equivalent and easier to obtain.

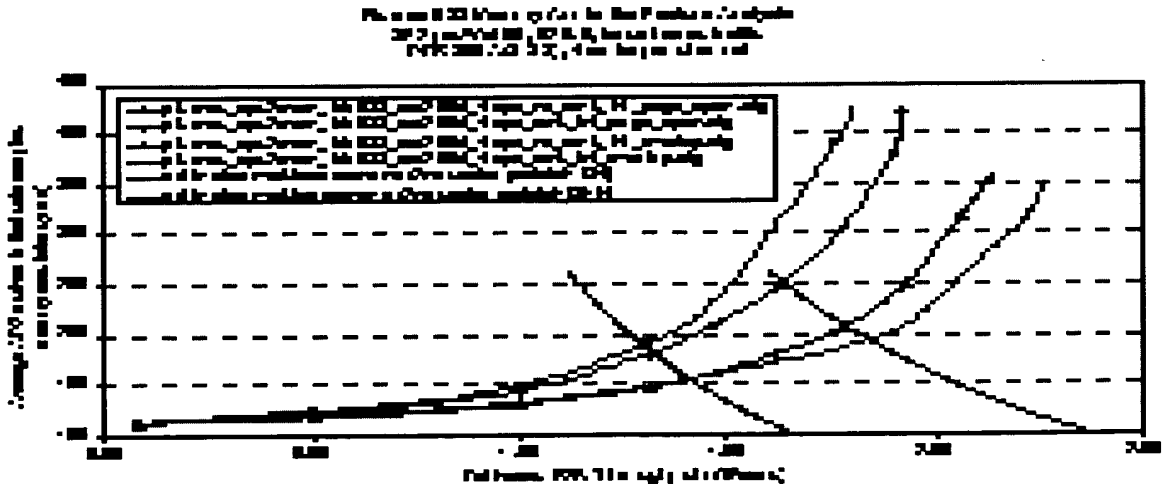
To investigate this further I used Little's law to estimate memory subsystem occupancy over a broad range of latency and bus utilization levels. The following table gives the memory subsystem occupancy predicted by Little's law for bus utilizations from 10% to 100% and latencies from 10 bus cycles to 30 bus cycles.

	Util									
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
latency	10	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
	15	0.75	1.50	2.25	3.00	3.75	4.50	5.25	6.00	6.75	7.50
	20	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
	25	1.25	2.50	3.75	5.00	6.25	7.50	8.75	10.00	11.25	12.50
	30	1.50	3.00	4.50	6.00	7.50	9.00	10.50	12.00	13.50	15.00

Table 1 Memory Subsystem Occupancy

So, for example, we could use a subsystem occupancy of 5 to achieve 100% of possible throughput when latency is low (10 bus clocks) without pre-fetch throttling. In addition this would throttle pre-fetching at higher latencies. To see where throttling would kick in at 20 bus clocks of latency, look across the row to an occupancy level of 5, then look to the top of the column. So pre-fetch throttling would start at 50% bus utilization when latency is 20 bus clocks. Or put another way (from another view point) at 50% bus utilization throttling begins when latency reaches 20 bus clocks.





Logic for DAMP:

The DAMP consists of two logic units: pre-fetch control logic and memory subsystem occupancy detection logic. When the microprocessor control register indicates that the multiple line pre-fetcher is enabled, the control logic will lookup the number of additional lines to fetch in a table. The table will have an entry for each level of pre-fetch. For example, if we want to be able to pre-fetch 0, 1, 2, or 3 lines we would use a 3 entry table. Each entry in the table has a valid bit. The memory subsystem occupancy detection logic dynamically adjusts the valid bits for each of the table entries to control the level of pre-fetch. The pre-fetch control logic uses the table to determine the number of lines to pre-fetch. If no entry is valid, there will be no pre-fetch. If there is a valid entry, that entry will indicate the number of lines to be pre-fetched. No more than one entry should every valid an one time.

Valid (1 bit)	**Number of Additional Sector** (2 bits)	**Subsystem Occupancy Deactivate** (N bits)	**Subsystem Occupancy Activate** (N bits)	**Notes**
1	**01**	**101**	**100**	**Level 3, Fetch 01h additional line**
0	**10**	**011**	**010**	**Level 2, Fetch 10h additional lines**
0	**11**	**001**	**000**	**Level 1, Fetch 11h additional lines**

Table 2 An example table used by DAMP.

The memory subsystem occupancy detection logic is a key component in DAMP. This logic tracts memory subsystem requests to generate an average occupancy for a given time interval. To generate the average occupancy, each bus clock the number of outstanding memory subsystem requests is added to a register.[1] At the end of the time interval this counter is divided by the time interval in bus clocks:[2]

$$\mu = (\Sigma n)/(\tau)$$

where μ is the average memory subsystem occupancy in the time interval 0 to τ. The occupancy detection logic uses μ to decide which, if any, valid bits in the DAMP table should be asserted. To determine which valid bit should be enabled the following decision tree is used:

Starting in the state where level N has its valid bit set,

if μ > deactivate level N then
    set valid bit for level N+1 and unset the valid bit for level N
else if μ < Activate level N-1 then
    set valid bit for level N-1 and unset the valid bit for level N
else
    set valid bit for level N (i.e., no change)

When μ is higher than the deactivate entry of the highest level in the table, level 3 in Table 2, no valid bit should be set. This configuration provides an element of hysteresis to the system to prevent the oscillation between any two levels.

Value to Intel:

The use of the DAMP mechanism promotes higher system performance through the use of a more effective pre-fetching mechanism. This technique works even better with SMP systems that have a greater potential for bus congestion. This mechanism will work nicely with existing and future core pre-fetchers[3] because DAMP will be driven, in part, by the core pre-fetcher but will adjust the amount of pre-fetching base on the memory subsystem

---

[1] An easy source of the number of outstanding requests is the depth of the IOQ. However, in systems which defer a significant number of bus transactions, the IOQ is a poor source of this information

[2] When the time interval is a power of two bus clocks there is no need for the divide operation. In fact not even a shift is needed since the significant bits can be used as is

[3] The first implementation will be in the DCP CFB which will have two cores, each with their own pre-fetch hardware

occupancy and hence be latency sensitive. Dual Core Potomac (DCP) plans to use the DAMP u-Architecture to improve SMP configuration performance (next generation Xeon server microprocessor).

In addition DAMP solves an outstanding problem we currently have with lack of scaling caused by overly aggressive pre-fetching. DAMP greatly reduces Intel's risk of having similar issues.

Prior Art and Novelty:

The closest prior art that I am aware of is that of the stream pre-fetch mechanism. There are many of those kinds of pre-fetchers that are based on either invoked instruction stream or data stream patterns, such as contents based pre-fetch, stride based pre-fetch, etc. Prescott and Tejas have the advanced pre-fetching mechanism in terms of instruction and data stream patterns. We are not aware any runtime dynamic and adaptive pre-fetch mechanism to optimize memory subsystem occupancy or access latency by adjusting pre-fetcher aggressiveness.

Detecting Infringement / Likely users:

Likely users include microprocessor and computer system companies, including Sun, IBM and AMD. Both microprocessor and computer system companies would be able to use this technology to improve their system performance. By checking run-time memory subsystem occupancy and micro-code implementation, we may indirectly detect the pre-fetching mechanism and its hardware supported dynamic adaptive ability.